By Lemma 1, the kissing number can be upper bounded by $\tau_n \le 2^{n(0.401+o(1))}$, making our bound, $F_n \le 2^{n(0.401+o(1))}$, a significant improvement in tightness over the previously known bounds mentioned above. The essential difference between our bound and the weaker one of Stone and Rogers is that theirs is based on a *minimal covering* while ours is based on a *maximal packing*.

Theorem 1: The maximum number of points in $\mathscr{R}^n$ which can have a common nearest neighbor is equal to the maximum kissing number (i.e., $F_n = \tau_n$), and is thus bounded as $2^{0.2075n(1+o(1))} \le F_n \le 2^{n(0.401+o(1))}$.

Proof: Consider any set of τ_n nonoverlapping spheres which are tangent to a common sphere. The centers of these τ_n spheres each have the center of the common sphere as a common nearest neighbor. Hence $F_n \ge \tau_n$ and it remains to show $F_n \le \tau_n$.

Suppose there exist $k > \tau_n$ points $\alpha_1, \cdots, \alpha_k$ which have the common touching point β. Let $d_m = \min\{d_{\alpha_i}: 1 \le i \le k\}$, where $d_{\alpha_i} = \|\alpha_i - \beta\|$ for each $i = 1, 2, \cdots, k$, and define

$$\gamma_i = \frac{d_{\alpha_i} - d_m}{d_{\alpha_i}}\beta + \frac{d_m}{d_{\alpha_i}}\alpha_i. \tag{4}$$

It is easily seen that

$$\|\gamma_i - \beta\| = \frac{d_m}{d_{\alpha_i}}\|\alpha_i - \beta\| = d_m. \tag{5}$$

The points $\gamma_1, \cdots, \gamma_k$ lie on a sphere in $\mathscr{R}^n$ of radius d_m and centered at β. Therefore, in the code $\{\beta, \gamma_1, \gamma_2, \cdots, \gamma_k\}$, each γ_i is a touching point of β with touching distance d_m.

We now show that for all i and j, if $i \ne j$, then $\|\gamma_i - \gamma_j\| \ge d_m$. Without loss of generality, suppose that i and j are indexes such that

$$\|\alpha_i - \beta\| \le \|\alpha_j - \beta\| \le \|\alpha_j - \alpha_i\|. \tag{6}$$

Then

$$\begin{aligned} 2(\alpha_i - \beta)\cdot(\alpha_j - \beta) &= \|\alpha_i - \beta\|^2 + \|\alpha_j - \beta\|^2 - \|\alpha_i - \alpha_j\|^2 \\ &\le d_{\alpha_i}^2 \le d_{\alpha_i} d_{\alpha_j}. \end{aligned} \tag{7}$$

But

$$2(\gamma_i - \beta)\cdot(\gamma_j - \beta) = \frac{2d_m^2}{d_{\alpha_i} d_{\alpha_j}}(\alpha_i - \beta)\cdot(\alpha_j - \beta) \tag{8}$$

by the definition of γ_i and γ_j.

Hence,

$$2(\gamma_i - \beta)\cdot(\gamma_j - \beta) \le d_m^2 \tag{9}$$

and

$$\begin{aligned} \|\gamma_i - \gamma_j\|^2 &= \|\gamma_i - \beta\|^2 + \|\gamma_j - \beta\|^2 \\ &\quad - 2(\gamma_i - \beta)\cdot(\gamma_j - \beta) \\ &\ge d_m^2 + d_m^2 - d_m^2 = d_m^2. \end{aligned} \tag{10}$$

Hence, in the code $\{\beta, \gamma_1, \gamma_2, \cdots, \gamma_k\}$, the point β has touching number k that exceeds τ_n. However, then k nonoverlapping spheres of radius $d_m/2$ could be placed about each touching point of β, violating the kissing number bound for the sphere centered at β of radius $d_m/2$. This completes the proof. The required bounds on F_n then follow from Lemma 1. □

Theorem 2: The average touching number of any Euclidean code in $\mathscr{R}^n$ is less than or equal to the maximum kissing number (i.e., $T \le \tau_n$), and thus is upper bounded by $2^{n(0.401+o(1))}$.

Proof: Given a Euclidean code, the directed *nearest-neighbor graph* associated with the code is defined in the following manner. Let each vertex correspond to a particular codepoint. A directed edge goes from vertex α to vertex β if β is a touching point of α. (If α is also a touching point of β, then there is another directed edge from β to α.)

From Theorem 1, it follows that the nearest-neighbor graph of any Euclidean code with $M > 1$ codepoints has the property that each vertex has at most τ_n incoming edges. Hence the total number of edges in the graph cannot exceed $M\tau_n$ (at most linear in the number of vertices) and consequently the total number of outgoing edges from all vertices is also at most $M\tau_n$. Therefore the average touching number T, being the average number of outgoing edges from a vertex, is bounded above by τ_n. The upper bound then follows from Lemma 1. □

References

[1] V. Klee, "On the complexity of d-dimensional Voronoi diagrams," *Arch. Math.*, vol. 34, pp. 75–80, 1980.

[2] K. Zeger and M. Kantorovitz, "Average number of facets per cell in tree-structured vector quantizer partitions," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 1053–1055, May 1993.

[3] J. Conway and N. J. A. Sloane, *Sphere Packings, Lattices, and Groups*. New York: Springer-Verlag, 1988.

[4] G. A. Kabatiansky and V. I. Levenshtein, "Bounds for packings on a sphere and in space," *Probl. Peredachi Inf.*, no. 1, pp. 3–25, 1978.

[5] A. D. Wyner, "Capabilities of bounded discrepancy decoding," *Bell Syst. Tech. J.*, pp. 1061–1122, 1965.

[6] A. Tversky, Y. Rinott, and C. M. Newman, "Nearest neighbor analysis of point processes: Applications to multidimensional scaling," *J. Math. Psychol.*, vol. 27, pp. 235–250, 1983.

[7] C. M. Newman, Y. Rinott, and A. Tversky, "Nearest neighbor and Voronoi regions in certain point processes," *Adv. Appl. Prob.*, vol. 15, pp. 726–751, 1983.

[8] L. T. Maloney, "Nearest neighbor analysis of point processes: Simulations and evaluations," *J. Math. Psychol.*, vol. 27, pp. 252–260, 1983.

[9] P. J. Bickel and L. Breiman, "Sums of functions of nearest neighbor distances, moment bounds, limit theorems and a goodness of fit test," *Ann. Prob.*, vol. 11, no. 1, pp. 185–214, 1983.

[10] L. P. Devroye and T. J. Wagner, "Distribution-free inequalities for the deleted and holdout error estimates," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 5, pp. 202–207, Mar. 1979.

[11] C. A. Rogers, "Covering a sphere with spheres," *Mathematika*, vol. 10, pp. 157–164, 1963.

[12] J. Fritz, "Distribution-free exponential error bound for nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-21, no. 5, pp. 552–557, Sept. 1975.

[13] C. J. Stone, "Consistent nonparametric estimation," *Ann. Stat.*, vol. 5, pp. 595–645, 1977.

Malfunction in the Peterson–Gorenstein–Zierler Decoder

Meera Srinivasan and Dilip V. Sarwate

***Abstract*—Most versions of the Peterson–Gorenstein–Zierler (PGZ) decoding algorithm are not true bounded distance decoding algorithms in the sense that when a received vector is not in the decoding sphere of**

Manuscript received March 8, 1993; revised October 22, 1993. The work of M. Srinivasan was supported by a National Science Foundation Graduate Fellowship. This paper was presented in part at the 1993 Conference on Information Sciences and Systems, Baltimore, MD, March 1993.

The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801.

IEEE Log Number 9404916.

any codeword, the algorithm does not always declare a decoding failure. For a t-error-correcting BCH code, if the received vector is at distance i, $i \le t$ from a codeword in a supercode with BCH distance $t + i + 1$, the decoder will output that codeword from the supercode. If that codeword is not a member of the t-error-correcting code, then decoder malfunction is said to have occurred. We describe the necessary and sufficient conditions for decoder malfunction, and show that malfunction can be avoided in the PGZ decoder by checking $t - \nu$ equations, where ν is the number of errors hypothesized by the decoder. A formula for the probability of decoder malfunction is also given, and the significance of decoder malfunction is considered for PGZ decoders and high-speed Berlekamp–Massey decoders.

***Index Terms*—Decoder malfunction, bounded-distance decoding, BCH coding, Reed–Solomon coding, Peterson–Gorenstein–Zierler algorithm.**

I. Introduction

A t-error-correcting bounded distance decoder decodes a received vector $\boldsymbol{r}$ into the unique codeword $\boldsymbol{c}$ at distance within distance t from $\boldsymbol{r}$ (if such a $\boldsymbol{c}$ exists), or declares failures if $\boldsymbol{r}$ is at distance greater than t from every codeword. It was generally believed that the well-known decoding algorithms for BCH codes, namely, the Peterson–Gorenstein–Zierler (PGZ) algorithm, the Berlekamp–Massey algorithm, and the Sugiyama–Kasahara–Hirasawa–Namekawa (SKHN) Euclidean algorithm [1]–[4], were bounded-distance decoding algorithms. However, Sarwate and Morrison [5] showed that PGZ and SKHN algorithms, along with the high-speed version of the Berlekamp–Massey algorithm, are subject to *decoder malfunction*, which they defined as the event in which the decoder does not declare failure when the received vector is not within the decoding sphere of any codeword, but instead produces something that is not a codeword at all. In this paper, we give a necessary and sufficient condition for malfunction in the PGZ decoder, and show that this malfunction can be avoided by checking $t - \nu$ equations, where ν is the number of errors hypothesized by the PGZ decoder. This test for avoiding malfunction was first obtained by Dür [6] using deep results from invariant theory of binary forms, but our proof uses only elementary linear algebra and is much simpler. The probability of decoder malfunction is then given for Reed–Solomon codes in order to evaluate the necessity of implementing the checks needed to avoid malfunction.

II. A Necessary and Sufficient Condition for Decoder Malfunction

Let $V_t(\boldsymbol{x})$ denote the set of vectors within Hamming distance t from the vector $\boldsymbol{x}$, and consider a code $\mathscr{C}$ with minimum distance $d \ge 2t + 1$. Then a t-error-correcting bounded distance decoder for $\mathscr{C}$ will output a codeword $\boldsymbol{c} \in \mathscr{C}$ if and only if the received vector $\boldsymbol{r}$ is in $V_t(\boldsymbol{c})$. If $\boldsymbol{r}$ is not in $V_t(\boldsymbol{c})$ for any $\boldsymbol{c} \in \mathscr{C}$, the decoder declares failure, i.e., it indicates that there is no codeword within distance t from $\boldsymbol{r}$. Let $\mathscr{C}^{(2t+1)}$ denote a t-error-correcting BCH code over GF(q) whose generator polynomial has zeros $\alpha, \alpha^2, \cdots, \alpha^{2t}$. If $c(x)$ is the transmitted codeword, $e(x)$ is the channel error polynomial, and $r(x) = c(x) + e(x)$ is the received polynomial, then the PGZ algorithm decodes $r(x)$ as follows [2], [4].

1) Compute the syndromes $S_i = r(\alpha^i) = e(\alpha^i)$ for $1 \le i \le 2t$.
2) Find the largest integer $\nu \le t$ such that the matrix

$$M_\nu = \begin{pmatrix} S_1 & S_2 & \cdots & S_\nu \\ S_2 & S_3 & \cdots & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} \end{pmatrix}$$

is nonsingular. Hypothesize that ν errors have occurred.

3) Solve the system

$$M_\nu[\lambda_\nu, \lambda_{\nu-1}, \cdots, \lambda_1]^T = -[S_{\nu+1}, S_{\nu+2}, \cdots, S_{2\nu}]^T$$

for the coefficients $\lambda_1, \lambda_2, \cdots, \lambda_\nu$ of the hypothesized error locator polynomial.

4) Factor the error locator polynomial $\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_\nu x^\nu$ into $\Lambda(x) = \prod_{i=1}^{\nu}(1 - X_\nu x)$. Then $X_1, X_2, \cdots, X_\nu$ are the error locations.

5) Solve the system

$$\begin{pmatrix} X_1 & X_2 & \cdots & X_\nu \\ X_1^2 & X_2^2 & \cdots & X_\nu^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^\nu & X_2^\nu & \cdots & X_\nu^\nu \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_\nu \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_\nu \end{pmatrix}$$

for the error values $Y_1, Y_2, \cdots, Y_\nu$.

6) The hypothesized error pattern is now

$$e^*(x) = Y_1 x^{\log_\alpha X_1} + Y_2 x^{\log_\alpha X_2} + \cdots + Y_\nu x^{\log_\alpha X_\nu}.$$

Correct the errors by subtracting $e^*(x)$ from $r(x)$ to obtain the corrected codeword.

If $e = wt[e(x)] \le t$, then it is known that $e^*(x) = e(x)$, and thus the PGZ decoder decodes the received vector correctly. When $e > t$, the decoder may fail to decode $r(x)$. If $e > t$ but $\boldsymbol{r} \in V_t(\boldsymbol{c})$ for some codeword in $\mathscr{C}$ then the decoder will still decode the received vector into a codeword, albeit erroneously. Otherwise, the decoder will generally declare failure when

- the error locations found in step 4 are not ν distinct nth roots of unity,
- some of the error values found in step 5 are zero or do not belong to the symbol field.

The phenomenon of decoder malfunction was studied by Sarwate and Morrison [5] who obtained the following sufficient condition for malfunction in PGZ decoders. Let $\mathscr{C} = \mathscr{C}^{(2t+1)}$ be a t-error-correcting BCH code, and let $\mathscr{C}^{(2t-2i+1)}$ denote the $(t - i)$-error-correcting supercode of $\mathscr{C}$ whose generator polynomial has roots $\alpha, \alpha^2, \cdots, \alpha^{2t-2i}$. Then, if $\boldsymbol{r} \in V_{t-2i}(\boldsymbol{c})$, $t \ge 2i$, where $\boldsymbol{c} \in \mathscr{C}^{(2t-2i+1)}$, the PGZ decoder output is $\boldsymbol{c}$. Since $\mathscr{C} \subset \mathscr{C}^{(2t-1)} \subset \cdots \subset \mathscr{C}^{(2t-2i+1)}$, it may happen that $\boldsymbol{c} \in \mathscr{C}$ also, in which case the decoding is correct if $e \le t$, and incorrect if $e > t$. On the other hand, if $\boldsymbol{c} \notin \mathscr{C}$, i.e., if $\boldsymbol{c} \in \mathscr{C}^{(2t-2i+1)} - \mathscr{C}$, then a decoder malfunction occurs.

The supercodes mentioned above have odd minimum distances $2t - 1, 2t - 3, \cdots, 2t - 2\lfloor t/2 \rfloor + 1$, respectively. The result of Sarwate and Morrison holds for the codes with even minimum distances $2t - 2, 2t - 4, \cdots, 2t - 2\lfloor t/2 \rfloor$ as well. These are codes that can correct $t - i$ errors and detect $t - i + 1$ errors, $i \le t/2$. If we combine the cases together, we obtain the following proposition. Although our results hold for BCH codes in general, we shall assume that we are using Reed–Solomon codes.

Proposition 1: For a t-error-correcting Reed–Solomon code $\mathscr{C} = \mathscr{C}^{(2t+1)}$ with minimum distance $2t + 1$, if $d(\boldsymbol{r}, \boldsymbol{c}) = i$ where $i \le t$ and $\boldsymbol{c} \in \mathscr{C}^{(t+i+1)}$, the output of the PGZ decoder is $\boldsymbol{c}$. If $\boldsymbol{c} \in \mathscr{C}^{(t+i+1)} - \mathscr{C}$, then malfunction occurs.

Proof: If $i = t$ in the above statement, then the received vector is within the decoding sphere of a codeword in $\mathscr{C}$ so the decoder will output that codeword as expected and we do not have a malfunction. But suppose that for $i \le t - 1, d(\boldsymbol{r}, \boldsymbol{c}) = i$ where $\boldsymbol{c} \in \mathscr{C}^{(t+i+1)}$. Let us consider what would happen if $\boldsymbol{r}$

were decoded using the decoder for $\mathscr{C}^{(t+i+1)}$. From the point of view of the decoder for $\mathscr{C}^{(t+i+1)}$, it is as though i errors have occurred. Now the code $\mathscr{C}^{(t+i+1)}$ can correct $\lfloor (t+i)/2 \rfloor$ errors and can detect $\lceil (t+i)/2 \rceil$ errors, so $\boldsymbol{r}$ is within the decoding distance of $\mathscr{C}^{(t+i+1)}$. The decoder for $\mathscr{C}^{(t+i+1)}$ will examine the matrices $M_{\lfloor (t+i)/2 \rfloor}, M_{\lfloor (t+i)/2 \rfloor - 1}, \cdots, M_{i+1}$ and find them to be singular, but will find M_i to be nonsingular, and thus would hypothesize that i errors have occurred. It would then solve

$$M_i[\lambda_i, \lambda_{i-1}, \cdots, \lambda_1]^T = -[S_{i+1}, S_{i+2}, \cdots, S_{2i}]^T$$

for the λ's and proceed to decode $\boldsymbol{r}$ into $\boldsymbol{c}$.

Since $d(\boldsymbol{r}, \boldsymbol{c}) = i$ and the code has minimum distance $t + i + 1$, it follows (see [4]) that the following $t + i - i = t$ equations hold:

$$S_j\lambda_i + S_{j+1}\lambda_{i-1} + \cdots + S_{j+i-1}\lambda_1 + S_{j+i} = 0, \qquad 1 \le j \le t.$$

Setting $j = 1, 2, \cdots, t$ in the above equations gives

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_i \\ S_2 & S_3 & \cdots & S_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_t & S_{t+1} & \cdots & S_{i+t-1} \end{pmatrix} \begin{pmatrix} \lambda_i \\ \lambda_{i-1} \\ \vdots \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} -S_{i+1} \\ -S_{i+2} \\ \vdots \\ -S_{i+t} \end{pmatrix}.$$

These t equations show that the $(i+1)$th row of M_t is a linear combination of the first i rows of M_t, and therefore, M_t, $M_{t-1}, \cdots, M_{i+1}$ are all singular. The matrices $M_t, M_{t-1}, \cdots, M_{\lfloor (t+i)/2 \rfloor + 1}$ are not examined by the decoder for $\mathscr{C}^{(t+i+1)}$, but they *are* examined by the decoder for our original code $\mathscr{C}$. The decoder for $\mathscr{C}$ finds $M_t, M_{t-1}, \cdots, M_{\lfloor (t+i)/2 \rfloor + 1}, M_{\lfloor (t+i)/2 \rfloor}, \cdots, M_{i+1}$ to be singular and M_i to be nonsingular. Thus, the decoder hypothesizes that i errors have occurred and goes through the same steps as the decoder for $\mathscr{C}^{(t+i+1)}$, thereby decoding $\boldsymbol{r}$ into $\boldsymbol{c}$, a codeword in $\mathscr{C}^{(t+i+1)}$. If $\boldsymbol{c} \in \mathscr{C}$ as well, the decoder succeeds in producing a valid codeword, whereas if $\boldsymbol{c} \notin \mathscr{C}$, malfunction occurs. □

It has been pointed out in [5] that the PGZ algorithm malfunctions in this way because not all of the syndrome values are used, i.e., S_{2t} is not used in determining the hypothesized number of errors ν that have occurred (step 2), and if the decoder hypothesizes that $\nu < t$, then S_{2t} is never used in the decoding process. One way to prevent malfunction from occurring is to simply add a step at the end of the algorithm that checks whether the output is a legitimate codeword. This would involve checking $2t$ equations to see if the codeword has the requisite roots.

Another way of preventing malfunction is to use the fact that malfunction cannot occur if the $2t - \nu$ equations

$$S_j\lambda_\nu + S_{j+1}\lambda_{\nu-1} + \cdots + S_{j+\nu-1}\lambda_1 + S_{j+\nu} = 0, \qquad 1 \le j \le 2t - \nu$$

hold [5]. Let T_j denote the quantity $S_j\lambda_\nu + S_{j+1}\lambda_{\nu-1} + \cdots + S_{j+\nu-1}\lambda_1 + S_{j+\nu}$. When the PGZ decoder hypothesizes that ν errors have occurred, it solves the system of ν equations in step 3 of the PGZ algorithm. Hence, the coefficients $\lambda_1, \lambda_2, \cdots, \lambda_\nu$ are such that T_j equals zero for $1 \le j \le \nu$. It follows that malfunction can be avoided by checking that the remaining $2t - 2\nu$ T_j, for $\nu + 1 \le j \le 2t - \nu$, are also zero. Dür [6] has used deep results from the invariant theory of binary forms to show that if the PGZ decoder hypothesizes that ν errors have occurred, then the coefficients $\lambda_1, \lambda_2, \cdots, \lambda_\nu$ found in step 3 are such that T_j equals zero for $1 \le j \le t$. Therefore, in order to avoid malfunction, the decoder need only check that the remaining T_j, $t + 1 \le j \le 2t - \nu$, are zero as well. An alternative proof of Dür's result using only elementary linear algebra follows.

Proposition 2: If the PGZ algorithm hypothesizes in step 2 that ν errors have occurred, then $T_j = 0$ for $1 \le j \le t$.

Proof: If ν errors are hypothesized by the decoder, then $|M_\nu| \neq 0$ and $|M_{\nu+1}| = |M_{\nu+2}| = \cdots = |M_t| = 0$. The PGZ decoder solves the system of ν equations in step 3, so that $T_j = 0$ for $1 \le j \le \nu$. We will show by induction that $T_j = 0$ for $\nu + 1 \le j \le t$ as well. If we premultiply the matrix $M_{\nu+1}$ by the matrix

$$A_{\nu+1} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ \lambda_\nu & \lambda_{\nu-1} & \cdots & \lambda_\nu & 1 \end{pmatrix},$$

we obtain

$$M^*_{\nu+1} = A_{\nu+1}M_{\nu+1} = \begin{pmatrix} S_1 & S_2 & \cdots & S_\nu & S_{\nu+1} \\ S_2 & S_3 & \cdots & S_{\nu+1} & S_{\nu+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} & S_{2\nu} \\ T_1 & T_2 & \cdots & T_\nu & T_{\nu+1} \end{pmatrix}$$

$$= \begin{pmatrix} S_1 & S_2 & \cdots & S_\nu & S_{\nu+1} \\ S_2 & S_3 & \cdots & S_{\nu+1} & S_{\nu+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} & S_{2\nu} \\ 0 & 0 & \cdots & 0 & T_{\nu+1} \end{pmatrix}$$

where the last line in the matrix follows from the fact that $T_j = 0$ for $1 \le j \le \nu$. We see that $M^*_{\nu+1}$ is of the form

$$\left(\begin{array}{c|c} M_\nu & B \\ \hline 0 & T_{\nu+1} \end{array}\right).$$

Hence, $|M^*_{\nu+1}| = |M_\nu|T_{\nu+1}$. But since $A_{\nu+1}$ is a lower triangular matrix with determinant 1, we also have that $|M^*_{\nu+1}| = |A_{\nu+1}||M_{\nu+1}| = |M_{\nu+1}| = 0$. Therefore, $|M_\nu|T_{\nu+1} = 0$, and since $|M_\nu| \neq 0$, it must be that $T_{\nu+1} = 0$.

Similarly, we show that $T_{\nu+2} = T_{\nu+3} = \cdots = T_t = 0$. Assume that $T_{\nu+1} = \cdots = T_{\nu+k-1} = 0$ for some $k \le t - \nu$. We want to show that $T_{\nu+k} = 0$. If we premultiply the matrix $M_{\nu+k}$ by the matrix

$$A_{\nu+k} = \begin{pmatrix} P_{\nu\times(\nu+k)} \\ Q_{k\times(\nu+k)} \end{pmatrix}$$

where $P_{\nu\times(\nu+k)} = (I_{\nu\times\nu}, 0_{\nu\times k})$ and the rows of $Q_{k\times(\nu+k)}$ are k right cyclic shifts of $[\lambda_\nu, \lambda_{\nu-1}, \cdots, \lambda_1, 1, 0, 0, \cdots, 0]$ of length $\nu + k$, we obtain

$$M^*_{\nu+k} = A_{\nu+k}M_{\nu+k}$$

$$= \left(\begin{array}{ccc|ccc} S_1 & \cdots & S_\nu & S_{\nu+1} & \cdots & S_{\nu+k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ S_\nu & \cdots & S_{2\nu-1} & S_{2\nu} & \cdots & S_{2\nu+k-1} \\ \hline T_1 & \cdots & T_\nu & T_{\nu+1} & \cdots & T_{\nu+k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ T_k & \cdots & T_{\nu+k-1} & T_{\nu+k} & \cdots & T_{\nu+2k-1} \end{array}\right)$$

$$= \left(\begin{array}{c|ccc|c} & S_{\nu+1} & \cdots & S_{\nu+k-1} & S_{\nu+k} \\ M_\nu & \vdots & \ddots & \vdots & \vdots \\ & S_{2\nu} & \cdots & S_{2\nu+k-2} & S_{2\nu+k-1} \\ \hline & 0 & \cdots & 0 & T_{\nu+k} \\ & 0 & \cdots & T_{\nu+k} & T_{\nu+k+1} \\ 0 & \vdots & \vdots & \ddots & \vdots \\ & T_{\nu+k} & \cdots & T_{\nu+2k-2} & T_{\nu+2k-1} \end{array}\right)$$

where the zeros follow from the induction hypothesis that $T_{\nu+1} = \cdots = T_{\nu+k-1} = 0$. The matrix in the lower-right quadrant is triangular with diagonal elements all $T_{\nu+k}$. Therefore,

$$|M^*_{\nu+k}| = |M_\nu|(-1)^{\lfloor k/2 \rfloor}(T_{\nu+i})^k.$$

Since $A_{\nu+k}$ is a lower triangular matrix with determinant 1, $|M^*_{\nu+k}| = |A_{\nu+k}||M_{\nu+k}| = |M_{\nu+k}| = 0$. Therefore,

$$|M_\nu|(-1)^{\lfloor k/2 \rfloor}(T_{\nu+i})^k = 0,$$

and since $|M_\nu| \neq 0$, we must have $T_{\nu+k} = 0$. Thus, we have shown that $T_{\nu+k} = 0$ for $1 \leq k \leq t - \nu$, i.e., we have shown that $T_j = 0$ for $1 \leq j \leq t$. □

We now show that the sufficient condition for the malfunction given in Proposition 1 is necessary as well. Consider the syndrome polynomial $S(x) = \sum_{j=1}^{2t} S_j x^{j-1}$ and the error-evaluator polynomial $\Omega(x) = \Lambda(x)S(x) \bmod x^{2t}$. It is known that if a BCH decoder decodes successfully and finds an error pattern of weight ν, then $\deg \Lambda = \nu$ and $\deg \Omega < \nu$ [2], [3], [5]. Using this fact, we prove the following proposition.

Proposition 3: If the PGZ decoder malfunctions and outputs a vector $\boldsymbol{c}$ as the most likely transmitted codeword, then $\boldsymbol{c} \in \mathscr{C}^{(t+\nu+1)} - \mathscr{C}$, where ν is the number of errors hypothesized by the PGZ decoder, and $\nu \leq t - 1$.

Proof: Suppose that the PGZ decoder malfunctions and produces an output vector $\boldsymbol{c}$ corresponding to an error pattern of weight ν and an error locator polynomial $\Lambda(x)$ of degree ν. Then by Proposition 2, it must be that $T_j = 0$ for $1 \leq j \leq t$. Now, it we consider the product $\Lambda(x)S(x)$, which is a polynomial of degree $\nu + 2t - 1$, the coefficients of $x^\nu, x^{\nu+1}, \cdots, x^{2t-1}$ are precisely $T_1, T_2, \cdots, T_{2t-\nu}$, respectively. Hence, we see that the coefficients of $x^\nu, x^{\nu+1}, \cdots, x^{\nu+t-1}$ are zero. However, the coefficients of $x^{\nu+t}, x^{\nu+t+1}, \cdots, x^{2t-1}$ cannot all be zero because if they were, then $\Lambda(x)S(x) = \Omega(x) \bmod x^{2t}$, where $\deg \Omega < \nu$, and therefore by [5], malfunction could not have occurred. Nonetheless, since the coefficients of $x^\nu, x^{\nu+1}, \cdots, x^{\nu+t-1}$ are zero in the product $\Lambda(x)S(x)$, we have that $\Lambda(x)S(x) \equiv \Omega(x) \bmod x^{\nu+t}$ where $\deg \Omega < \nu$. This corresponds to a successful decoding for $\mathscr{C}^{(t+\nu+1)}$ in which the decoder finds a codeword $\boldsymbol{c} \in \mathscr{C}^{(t+\nu+1)}$ at distance ν from $\boldsymbol{r}$. Since decoder malfunction did occur, it must be that $\boldsymbol{c} \in \mathscr{C}^{(t+\nu+1)} - \mathscr{C}$. Finally, note that if $\nu = t$, then $T_1 = 0, \cdots, T_t = 0$ are the coefficients of $x^t, \cdots, x^{2t-1}$. Therefore, $\Lambda(x)S(x) \equiv \Omega(x) \bmod x^{2t}$, where $\Omega(x)$ has degree less than t, so malfunction cannot occur. Thus, we have shown that if the PGZ decoder malfunctions with ν errors and outputs a vector $\boldsymbol{c}$, then $\boldsymbol{c} \in \mathscr{C}^{(t+\nu+1)} - \mathscr{C}$ and $\nu \leq t - 1$. □

Let us comment here that we have stated our results for Reed–Solomon codes because $\mathscr{C}^{(d)}, \mathscr{C}^{(d-1)}, \cdots$ are all distinct in that case. The results are applicable to general BCH codes as well, but they must be interpreted carefully because the codes $\mathscr{C}^{(d)}, \mathscr{C}^{(d-1)}, \cdots$ may not all be distinct in the general case. Also, note that a special case of malfunction has been accounted for previously. Some implementations of the PGZ decoder declare failure if the matrices $M_t, M_{t-1}, \cdots, M_1$ are all found to be singular even though the syndromes are not all zero. If $M_t, M_{t-1}, \cdots, M_1$ are all singular, then the PGZ algorithm as described above hypothesizes that $\nu = 0$ and finds $\Lambda(x) = 1$. Therefore by Proposition 2, $S_1 = S_2 = \cdots = S_t$ must be zero, and the test for malfunction (does $T_j = 0$ for $t + 1 \leq j \leq 2t - \nu$?) reduces to checking whether $S_{t+1}, S_{t+2}, \cdots, S_{2t}$ are zero, which is essentially a check that is already performed in some implementations of the PGZ algorithm.

As was pointed out in [5], Chen's high-speed version of the Berlekamp–Massey decoding algorithm [7] can malfunction because it achieves it high speed by not checking whether $T_j = 0$ for $t + 1 \leq j \leq 2t - \nu$. Proposition 2 essentially explains why this decoding algorithm malfunctions. When this algorithm finds a $\Lambda(x)$ of degree ν, it checks whether $T_{\nu+1}, T_{\nu+2}, \cdots, T_t$ are all zero, and if so, it does not check whether the remaining T_j, $t + 1 \leq j \leq 2t - \nu$ are also zero. Therefore, when the received vector is at distance ν from a codeword in $\mathscr{C}^{(t+\nu+1)}$, the high-speed decoder finds $\Lambda(x)$ of degree ν, and since Proposition 2 guarantees that $T_j = 0$ for $1 \leq j \leq t$, the decoder skips the remaining checks, and thus malfunctions in exactly the same way that the PGZ decoder does. It follows that Propositions 1 and 3 are also applicable to Chen's high-speed version of the Berlekamp–Massey decoding algorithm.

III. The Probability of Decoder Malfunction

Because testing for malfunction does require additional hardware or software instructions, it is of some practical importance to evaluate the probability of decoder malfunction in order to determine its significance in increasing the probability of incorrect decoding. In the previous section, we showed that malfunction occurs if and only if the received vector $\boldsymbol{r}$ is such that $d(\boldsymbol{r}, \boldsymbol{c}) = i$, $i \leq t - 1$, where $\boldsymbol{c} \in \mathscr{C}^{(t+i+1)} - \mathscr{C}$. Equivalently, since the code is linear, malfunction occurs if and only if the error pattern $\boldsymbol{e}$ is such that $d(\boldsymbol{e}, \boldsymbol{c}) = i$, $i \leq t - 1$, where $\boldsymbol{c} \in \mathscr{C}^{(t+i+1)} - \mathscr{C}$. Hence

$$P\{\text{malfunction}\} = P_{mf} = \sum_{i=0}^{t-1} P[d(\boldsymbol{e}, \boldsymbol{c}) = i | \boldsymbol{c} \in \mathscr{C}^{(t+i+1)} - \mathscr{C}]$$

$$= \sum_{i=0}^{t-1} P[d(\boldsymbol{e}, \boldsymbol{c}) = i | \boldsymbol{c} \in \mathscr{C}^{(t+i+1)}] - P[d(\boldsymbol{e}, \boldsymbol{c}) = i | \boldsymbol{c} \in \mathscr{C}]. \tag{1}$$

In contrast, the probability of undetected error is given by

$$P_e = \sum_{i=0}^{t} P[d(\boldsymbol{e}, \boldsymbol{c}) = i | \boldsymbol{c} \in \mathscr{C}^{(2t+1)} - 0].$$

We assume that our q-ary (n, k) Reed–Solomon code $\mathscr{C}$ is used on a discrete memoryless channel with q inputs and q outputs, and that any transmitted symbol has a probability $\epsilon/(q-1)$ of being changed into each of the $q - 1$ other symbols. Let $A_l^{(d)}$ be the number of codewords of weight l in $\mathscr{C}^{(d)}$. The weight enumerator polynomial for $\mathscr{C}^{(d)}$ is $A^{(d)}(z) = \sum_{l=0}^{n} A_l^{(d)} z^l$. Let us also define $B_l(i, k)$ to be the number of vectors of weight k at distance i from a given vector of weight l. Then the probability that an error pattern is at distance i from a codeword in $\mathscr{C}^{(d)}$ is

given by [2], [8]

$$P[d(\boldsymbol{e},\boldsymbol{c}) = i | \boldsymbol{c} \in \mathscr{C}^{(d)}] = \sum_{k=0}^{n} \sum_{l=0}^{n} \left(\frac{\epsilon}{q-1}\right)^{k} (1-\epsilon)^{n-k} A_l^{(d)} B_l(i,k).$$

Substituting this into (1), the probability of malfunction is

$$P_{mf} = \sum_{i=0}^{t-1} \sum_{k=0}^{n} \sum_{l=0}^{n} \left(\frac{\epsilon}{q-1}\right)^{k} (1-\epsilon)^{n-k} \cdot [A_l^{(t+i+1)} - A_l^{(2t+1)}] B_l(i,k). \quad (2)$$

Since the weight enumerators of Reed–Solomon codes are known, it is easy to compute P_{mf} for these codes. The weight distribution of a length n RS code over GF(q) with minimum distance d is given by [2], [3] $A_0^{(d)} = 1$, $A_l^{(d)} = 0$, for $1 \le l \le d-1$, and

$$A_l^{(d)} = \binom{n}{l}(q-1) \sum_{j=0}^{l-d} (-1)^j \binom{l-1}{j} q^{l-d-j} \quad \text{if } l \ge d. \quad (3)$$

The quantity $B_l(i,k)$ can be computed for any values of l, i, and k using the following formula [2]:

$$B_l(i,k) = \sum_{\substack{0 \le \eta \le n \\ 0 \le \nu \le n \\ \eta + 2\nu + k = i + l}} \binom{n-l}{\nu + k - l} \binom{l}{\eta} \binom{l-\eta}{\nu} \cdot (q-1)^{\nu+k-l} (q-2)^{\eta}. \quad (4)$$

Using (2)–(4), P_{mf} can be evaluated for any Reed–Solomon code over GF(q) used on a discrete memoryless channel with symbol error probability ϵ.

Since (4) is complicated to evaluate, simple bounds on P_{mf} are of interest. When the symbol error probability ϵ is very small, a decoder error or malfunction occurs primarily because of error patterns of weight $t+1$. But for $t+1 \le d \le 2t+1$, there are $\binom{d}{t+1}$ error patterns of weight $t+1$ at distance $d-t-1$ from a given codeword of weight d in $\mathscr{C}^{(d)}$, and there are $\binom{n}{d}(q-1)$ codewords of weight d in $\mathscr{C}^{(d)}$. Hence,

$$P_e \propto \binom{2t+1}{t+1}\binom{n}{2t+1}(q-1)$$

and

$$P_{mf} \propto \sum_{d=t+1}^{2t} \binom{d}{t+1}\binom{n}{d}(q-1).$$

Therefore,

$$\frac{P_{mf}}{P_e} \approx \sum_{i=0}^{t-1} \frac{t(t-1)\cdots(t-i)}{(n-2t)(n-2t+1)\cdots(n-2t+i)}$$

as ϵ approaches 0. The above sum is obviously bounded below by the first summand $t/(n-2t)$ and can be bounded above by the sum of a geometric series with ratio $t/(n-2t)$. If $n > 3t$, this ratio is less than one, and hence

$$\frac{t}{n-2t} < \frac{P_{mf}}{P_e} < \frac{t}{n-3t} \quad \text{if } n > 3t. \quad (5)$$

If $n \le 3t$, it is easily seen that $P_{mf} > P_e$. As an application of (5), note that as $\epsilon \to 0$, the ratio P_{mf}/P_e for a $(32,24)$ Reed–Solomon code over GF(2^5) lies between $1/6$ and $1/5$ (the actual value is 0.188), while for a $(32,12)$ Reed–Solomon code over GF(2^5), P_{mf}/P_e lies between $5/6$ and 5 (the actual value is 1.97).

IV. Discussion

The effect of malfunction on the performance of a decoder is to increase the probability that the output vector of the decoder is a vector other than the transmitted codeword. Thus, the probability that the output is not the transmitted codeword increases from P_e to $P_e + P_{mf}$. From (5), it appears that the effective increase in error probability is small if t is relatively small compared to the block length n, but the increase can be substantial for low rate codes. Now, the event of undetected decoder error is not preventable, but the event of decoder malfunction is very definitely preventable. However, checks to avoid malfunction require additional hardware or software instructions, which increase the cost of implementation and the running time. Therefore, as a practical matter, the system designer must consider the tradeoff between reduction of the probability of incorrect decoding and the cost of preventing decoder malfunction.

The quantities P_{mf} and P_e have been calculated exactly as functions of the symbol error probability ϵ for some Reed–Solomon codes of blocklength 32 [9]. As expected from (5), P_{mf} is comparable to P_e for the lower rate codes and much smaller than P_e for the higher rate codes. Therefore, a noticeable improvement in the probability of incorrect decoding may be achieved in low rate codes if a check is included in the algorithm to prevent malfunction. Of course, in practice, the PGZ algorithm is typically used only for decoding high rate codes. In such cases, P_{mf} is relatively small compared to P_e, and hence the additional expense of implementing checks to prevent malfunction may not be merited. However, as pointed out in Section II, certain high-speed Berlekamp–Massey decoders malfunction in exactly the same manner as the PGZ decoder because these decoders omit certain checks, and these checks are precisely the ones needed to detect cases of impending decoder malfunction. Berlekamp–Massey decoders are typically used for low rate codes, and since the computational burden increases with the error-correcting capability, one would be most tempted to use the high-speed version of the Berlekamp–Massey algorithm when the code rate is low. This is exactly the situation in which the malfunction probability is significant compared to the error probability, and in which the tradeoff between decoding speed and increase in the probability of incorrect decoding must be given serious consideration.

References

[1] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[2] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

[3] F. J. MacWilliams and N. J. A. Sloane, *Theory of Error Correcting Codes*. Amsterdam: North-Holland, 1977.

[4] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: M.I.T. Press, 1972.

[5] D. V. Sarwate and R. D. Morrison, "Decoder malfunction in BCH decoders," *IEEE Trans. Inform. Theory*, vol. 36, pp. 884–889, July 1990.

[6] A. Dür, "Avoiding decoder malfunction in the Peterson–Gorenstein–Zierler decoder," *IEEE Trans. Inform. Theory*, vol. 39, pp. 640–643, Mar. 1993.

[7] C. L. Chen, "High-speed decoding of BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 254–256, Mar. 1981.

[8] T. Kasami and S. Lin, "On the probability of undetected error for the maximum distance separable codes," *IEEE Trans. Commun.*, vol. COM-32, pp. 998–1006, Sept. 1984.

[9] M. Srinivasan, "Malfunction in the Peterson–Gorenstein–Zierler decoder," Master's thesis, Dep. Elec. Comput. Eng., Univ. Illinois, Urbana–Champaign, Jan. 1993.